

Motion Planning Strategies for Autonomously Mapping 3D Structures

Manikandasriram Srinivasan Ramanagopal, *Student Member, IEEE*, and Jerome Le Ny, *Member, IEEE*

Abstract—This paper presents a system capable of autonomously mapping the visible part of a bounded three-dimensional structure using a mobile ground robot equipped with a depth sensor. We describe motion planning strategies to determine appropriate successive viewpoints and attempt to fill holes automatically in a point cloud produced by the sensing and perception layer. We develop a local motion planner using potential fields to maintain a desired distance from the structure. The emphasis is on accurately reconstructing a 3D model of a structure of moderate size rather than mapping large open environments, with applications for example in architecture, construction and inspection. The proposed algorithms do not require any initialization in the form of a mesh model or a bounding box. We compare via simulations the performance of our policies to the classic frontier based exploration algorithm. We illustrate the efficacy of our approach for different structure sizes, levels of localization accuracy and range of the depth sensor.

Note to Practitioners—The objective of this work is to automate the process of building a 3D model of a structure of interest that is as complete as possible, using a mobile depth sensor, in the absence of any prior information about this structure. Given that several robust solutions for the Visual Simultaneous Localization and Mapping problem (vSLAM) are readily available, the key challenge that we address here is to develop motion planning policies to control the trajectory of the sensor. We present a system that is capable of autonomously determining the boundaries of the structure, before attempting to fill the holes in the constructed model. The performance of our system is illustrated through simulations performed with a depth sensor carried by a mobile manipulator.

Index Terms—Motion Planning, Active Sensing, Active SLAM, Autonomous Mapping, Autonomous Inspection

I. INTRODUCTION

ACCURATE 3D computer models of large structures have a wide range of practical applications, from inspecting an aging structure to providing virtual tours of cultural heritage sites [1], [2]. In order to autonomously build such a 3D model in real-time, we need to address two problems. First, we need a robust mapping system that can build the 3D model on the fly when given a sequence of images or depth maps as input. This is a widely researched problem called Visual Simultaneous Localization and Mapping (vSLAM), for which several open source packages offer increasingly accurate and efficient solutions [3], [4]. The second problem relates to

active sensing [5], as we need motion planning strategies that can guide a mobile sensor to explore the structure of interest. For mapping, monitoring or inspection applications, certain classical strategies such as frontier-based exploration algorithms [6], which guide the robot to previously unexplored regions irrespective of whether it is part of the structure of interest or not, are not necessarily well adapted.

The goal of this work is to guide a mobile ground robot equipped with a depth sensor, in order to autonomously determine the boundaries of an initially unknown structure, build a 3D model of the structure and attempt to fill holes in the model so that the reconstruction is as accurate and complete as possible. Some recent work considers the problem of reconstructing a 3D model of arbitrary objects by moving a depth sensor relative to the object [7], [8]. Typically, these systems iteratively build a complete 3D model of the object by heuristically choosing the next best viewpoint according to some performance measure. However, much of this work is restricted to building models of relatively small objects that are bounded by the size of the robot workspace. In contrast, our focus is on 3D reconstruction of larger but still bounded structures such as buildings, which can be several orders of magnitude larger than a mobile robot. The related problem of automated inspection deals with large structures such as tall buildings [9] and ship hulls. Bircher et al. [10] assume that a prior 3D mesh of the structure to inspect is available and compute a short path connecting viewpoints that together are guaranteed to cover all triangles in the mesh. As they point out, the inspection problem starting from a prior model is related to coverage path planning, see, e.g., [11], [12]. In [13], Englot et al. begin by assuming a safe bounding box of the hull and construct a coarse mesh of the hull by tracing along the walls of this box in a fixed trajectory without taking feedback from the actual geometry of the structure. Moreover, this coarse mesh is manually processed offline to yield an accurate 3D mesh which is then used to inspect the finer structural details. Sheng et al. [14] use a prior CAD model of an aircraft to plan a path for a robotic crawler such that it inspects all the rivets on the surface of the aircraft. In this paper however, we do not assume any prior information in terms of a 3D mesh, CAD model or a bounding box around the structure, and focus on reactive path-planning to build the model online.

In computer vision and photogrammetry, Structure from Motion (SfM) techniques aim at building a 3D model of a scene from a large number of images [16]–[18]. But most of this work focuses on batch post-processing and in any case assumes a given dataset. On the other hand, our work focuses on actively exploring the environment to build a complete

Part of this work was performed while the first author was visiting Polytechnique Montreal, under a Globalink Fellowship from MITACS. This work was also supported by NSERC (Grant 435905-13) and the Canada Foundation for Innovation (Grant 32848).

M. S. R. is with the Department of Electrical Engineering, IIT Madras, Chennai, India. J. Le Ny is with the Department of Electrical Engineering, Polytechnique Montreal, and GERAD, Montreal, QC H3T 1J4, Canada. e-mails: srmanikandasriram@gmail.com, jerome.le-ny@polymtl.ca.



(a)



(b)

Fig. 1. Comparison of the a) Simulated Model in Gazebo [15] that needs to be mapped and b) Reconstructed 3D model by a mobile ground robot using our policies. Only the bottom portion is mapped due to the limited reachable space of the sensor.

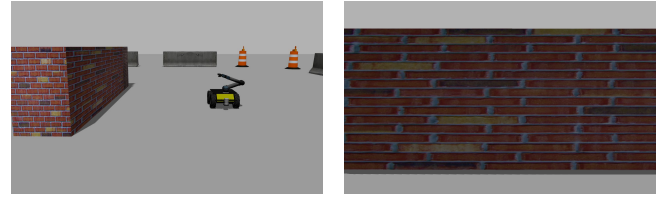
model in real-time, with our controller taking at any time the current model as an input. Naturally, eventual completeness of the model can be limited by the physical characteristics of the robot, and specifically the reachable space of the sensor, see Fig. 1. We emphasize that we do not discuss in details the task of actually building the model from the collected depth maps, which can be executed by one of the available vSLAM systems, such as the Real-Time Appearance Based Mapping package (RTAB-Map) [3] that we use in our simulations. This package can in fact be replaced with little change to our algorithms by any vSLAM system based on pose-graph optimization [19]. State-of-the-art SfM systems can also be used to post-process the sequence of images or depth maps captured using our policies in order to obtain a more accurate model.

Finally, another line of work in informative path planning relates to autonomous exploration and coverage of relatively large environments, using variants of frontier based exploration algorithms for example [20]–[23]. While these papers focus on path planning to quickly build models of potentially large and complex spaces, they do not address the problem of autonomously delimiting and mapping as completely as possible a specific bounded structure of interest.

In summary, the key contributions of this work are:

- a motion planning strategy to autonomously determine the boundaries of an unknown structure using a ground robot;
- a novel algorithm to determine incomplete portions of the partially constructed model;
- motion planning policies for automatically exploring and adding these missing portions to the model;
- and an evaluation of the proposed policies via simulations.

The rest of the paper is structured as follows. We begin with a detailed presentation of the problem in Section II. In Section III we present our policies for autonomously determining the boundaries of the unknown structure. Section IV describes algorithms for detecting the missing portions and completing the model. In Section V, we evaluate the proposed policies via simulations and present a comparison with the classic frontier based exploration algorithm. Finally, we discuss avenues for future work and conclude in Section VI.



(a)



(b)

Fig. 2. The starting configuration of the robot needs to satisfy Initial Conditions 1 and 2. a) At the beginning, the structure and the robot can be separated by a plane. b) The initial image as seen by the camera. Initially, the robot only knows that the structure in the FOV of its camera is the one that should be mapped.

II. PROBLEM STATEMENT AND ASSUMPTIONS

Consider the problem of constructing a 3D model of a given structure of finite size, such as a monument or a building for example. Initially, no approximate model of the structure nor map of the environment is available, and the actual size of the structure is also unknown. We address the following question: “How should a mobile robot carrying a depth sensing camera, such as a Kinect, move in order to reconstruct a complete 3D model of this structure?”. The sensor collecting depth and luminance images of the scene allows the robot to build the 3D model on the fly using available SLAM algorithms, such as RTAB-Map [3] or RGBD-SLAM [4]. These algorithms assemble the sequence of points clouds captured by the sensor (also called camera in the following), producing a registered global point cloud or a 3D occupancy grid stored in an OctoMap [24]. Note that the type of sensor (monocular camera, stereo camera rig with IMU, etc...) used depends on the SLAM algorithm. Any of these algorithms can be used with our policies as long as the SLAM module can additionally return the sequence of point clouds and corrected camera positions following registration. The remaining problem that we consider here is to determine the trajectory of the camera such that the entire visible portion of the structure is eventually captured in the model. The key challenge is to develop strategies that are applicable for any type of structure while respecting the physical limitations of the platform.

In order to specify to our system which structure is to be mapped, we require that the initial configuration of the robot

with respect to the structure satisfy the following two basic conditions, illustrated in Fig. 2.

Initial Condition 1: The robot is placed fully outside the structure, so that there exists a 2D plane separating the robot and the structure.

Initial Condition 2: The structure to be mapped is present in the field-of-view (FOV) of the camera.

Next, note that depending on the set of all configurations that are reachable by the mobile sensor mounted on a specific physical platform, some parts of the structure might not be visible at all, and hence cannot be mapped by any algorithm implemented on this platform. For concreteness, we make the following assumption to describe our scenario and algorithms, but other situations could be handled with the generic tools developed in this paper.

Assumption 1: The camera is mounted on a mobile ground robot such that the camera center C lies directly above the center R of the robot's base Frame of Reference (FoR), at a constant height. Moreover, the relative pitch and roll of the camera with respect to the robot FoR are kept fixed, while the relative yaw is unconstrained.

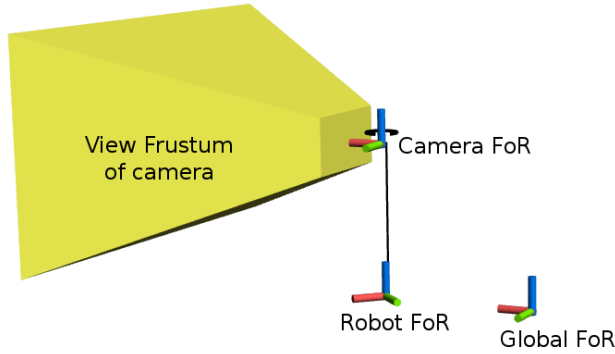


Fig. 3. The camera is kept at a constant height above the robot's base. The red, green and blue lines correspond to the x , y and z axes respectively and both the camera and robot can rotate along their z axes. The yellow region corresponds to the view frustum of the camera.

Based on Assumption 1, Fig. 3 shows our conventions for the different FoR used. The global point cloud is assembled in a global FoR $G\mathbf{x}_g\mathbf{y}_g\mathbf{z}_g$. Note that we use bold font to represent vectors. The robot FoR $R\mathbf{x}_r\mathbf{y}_r\mathbf{z}_r$ (forward, left, up) is attached to a point R on the base of the mobile robot and moves along with it. The camera FoR $C\mathbf{x}_c\mathbf{y}_c\mathbf{z}_c$, with $\mathbf{z}_c = \mathbf{z}_r$, is rigidly attached to the robot except for the yaw motion, which is left unconstrained. The imaging plane of the camera is defined by $\mathbf{y}_c\mathbf{z}_c$, with \mathbf{x}_c pointing towards the front of the camera on the optical axis. Coordinates in the camera, robot and global FoR are denoted using superscripts as \mathbf{v}^c , \mathbf{v}^r and \mathbf{v}^g respectively for a vector \mathbf{v} .

We make two additional assumptions for simplicity of exposition. First, let ρ be the maximum distance that we wish to allow between the structure and the camera when capturing point clouds. This distance could be the range of the camera or a shorter distance for which the resolution is higher. The next assumption guarantees that there exists collision free paths around the structure.

Assumption 2: The distance of the closest obstacle from the structure is at least 2ρ .

The next assumption simplifies the problem of detecting, tracking and removing the ground surface from point clouds.

Assumption 3: The structure and the robot are placed on a horizontal surface (so $\mathbf{z}_c = \mathbf{z}_r = \mathbf{z}_g$).

A consequence of these assumptions is that relatively horizontal surfaces that are at the same height or above the camera center cannot be mapped, and the maximum height of the structure that can be mapped is $H_{\max} = z_c^g + \rho \tan \psi / 2$, where ψ is the vertical angle of view of the camera and z_c^g the height of the camera. Assumption 3 could be removed by using recent classification systems that can differentiate between ground and non-ground regions [25] to pre-process the point clouds before sending them to our system.

Finally, there are additional implicit assumptions that we state informally. First, since we rely on an external mapping module to build the 3D model, the conditions that allow this module to operate sufficiently reliably must be met. For example, vSLAM generally requires appropriate scene illumination and the presence of a sufficiently rich set of visual features. Second, we concentrate on the reconstruction of the details of the model at a scale comparable with or larger than the typical length of the robot. If features at a smaller scale need to be included, e.g., fine structural details on a wall, our system could be augmented with a more local planner for a robotic arm carrying the sensor [7], [26], as well as targeted computer vision techniques [18]. Finally, for reasons explained in Section III-C, we assume that the robot is equipped with sensors capable of detecting obstacles in a 180° region ahead of it and within a distance of ρ .

We divide our mapping process into two phases. The first phase is the Perimeter Exploration (PE) phase, during which the robot moves clockwise around the structure to determine its boundaries. The robot continuously moves towards previously unseen regions of the structure, with the exploration directed towards finding the limits of the structure rather than closely following its geometry. The PE phase ends when our algorithm detects that the robot has returned to the neighborhood of its starting point O and the vSLAM module detects a global loop closure. After completing the PE phase, the system determines the locations of potential missing parts in the constructed 3D model. We can then start the second phase, which we call the Cavity Exploration (CE) phase, during which the system explores these missing parts in the model. The following subsections explain each step of our process in detail.

III. PERIMETER EXPLORATION

In this section, we present our first contribution - a method to autonomously determine the boundaries of an unknown structure. From Assumptions 2 and 3, $z_c^g = 0$ and $z_c^g = H_{\max}$ are bounding horizontal planes for the model. The remaining problem is to determine the expansion of the structure in the $\mathbf{x}_g\mathbf{y}_g$ plane. To do this, the robot moves clockwise around the structure by determining online a discrete sequence of successive goals or waypoints. It tries to keep the optical

Algorithm 1 Algorithm for computing the next goal for the camera using the current point cloud in camera FoR.

```

1: function COMPUTENEXTGOAL(cloud_full)
2:   cloud  $\leftarrow$  PCLremoveGroundPlane(cloud_full)
3:   cloud_slice  $\leftarrow$  filterForwardSlice(cloud)
4:    $\bar{p}^c \leftarrow$  PCLcompute3Dcentroid(cloud_slice)
5:    $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3; \lambda_1, \lambda_2, \lambda_3] \leftarrow$  PCA(cloud_slice)
6:    $\tilde{\mathbf{n}} \leftarrow \mathbf{v}_3 - (\mathbf{v}_3 \cdot \mathbf{z}_c) \mathbf{v}_3 \triangleright$  Projection on the  $\mathbf{x}_c \mathbf{y}_c$  plane
7:    $\mathbf{n} \leftarrow \tilde{\mathbf{n}} \text{ sign}(\tilde{\mathbf{n}} \cdot \overrightarrow{C\bar{p}^c}); \mathbf{n} \leftarrow \mathbf{n} / \|\mathbf{n}\|$ 
8:    $\mathbf{r} \leftarrow \mathbf{z}_c \times \mathbf{n}$ 
9:   goal  $\leftarrow \bar{p}^c - D \mathbf{n} + \text{step} \mathbf{r}$ 
10:  return goal,  $\mathbf{n}$ 
11: end function

```

axis of the depth sensor approximately perpendicular to the structure, which maximizes the depth resolution at which a given portion of the structure is captured, and increases the density of captured points. It also tries to maintain the camera center C on a smooth path at a fixed distance from the structure.

A. Determination of the next goal

The pseudo-code to determine the next position and orientation of the camera in our perimeter exploration algorithm is shown in Algorithm 1. The algorithm takes as input the current point cloud produced by the camera in its FoR. For its implementation we rely on the Point Cloud Library (PCL) [27]. First, the ground plane is removed so that the resulting point cloud \mathcal{P} contains only those points that belong to the structure. Next, on line 3, we select a subset \mathcal{S} of the point cloud referred to as the forward slice, which adjoins the part of the structure that must be explored next, see Fig. 4. Concretely, we choose \mathcal{S} so that its y^c -coordinates satisfy $y_{\max}^c - \frac{y_{\max}^c - y_{\min}^c}{3} \leq y^c \leq y_{\max}^c$, where y_{\min}^c and y_{\max}^c are the minimum and maximum y^c -coordinate values for all points in \mathcal{P} . On line 5, following [28], we compute via Principal Component Analysis (PCA) the normal direction to that plane Π which best fits \mathcal{S} . In more details, denote $\mathcal{S} = \{p_i^c : i = 1, 2, \dots, m\}$ and define the covariance matrix $\mathbf{X} = \frac{1}{m} \sum_{i=1}^m (p_i^c - \bar{p}^c)(p_i^c - \bar{p}^c)^T$, where $\bar{p}^c = \frac{1}{m} \sum_{i=1}^m p_i^c$ is the centroid of \mathcal{S} computed on line 4. We compute the eigenvectors $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ of \mathbf{X} , ordered here by decreasing value of the eigenvalues $\lambda_1, \lambda_2, \lambda_3$. The eigenvector \mathbf{v}_3 for the smallest eigenvalue corresponds to the normal to the plane Π .

The algorithm returns \mathbf{n} , computed from the projection of the normal vector \mathbf{v}_3 on the $\mathbf{x}_c \mathbf{y}_c$ plane, and taken to point in the direction of the vector $\overrightarrow{C\bar{p}^c}$. This vector \mathbf{n} defines the desired orientation of the camera. The algorithm also returns the next goal point *goal* $= \bar{p}^c - D \mathbf{n} + \text{step} \mathbf{r}$ for the center C of the camera, where $D < \rho$ is the desired distance between the camera and the structure, $\mathbf{r} = \mathbf{z}_c \times \mathbf{n}$ is computed on line 8, and $\text{step} = \frac{y_{\max}^c - y_{\min}^c}{6}$. The term $\text{step} \mathbf{r}$, which is along the plane Π , is used to shift the *goal* forward so that both sections of a corner fall in the FOV of the camera, as in the situation shown on Fig. 4. This prevents the algorithm from making slow progress around corners. Finally, the computed camera

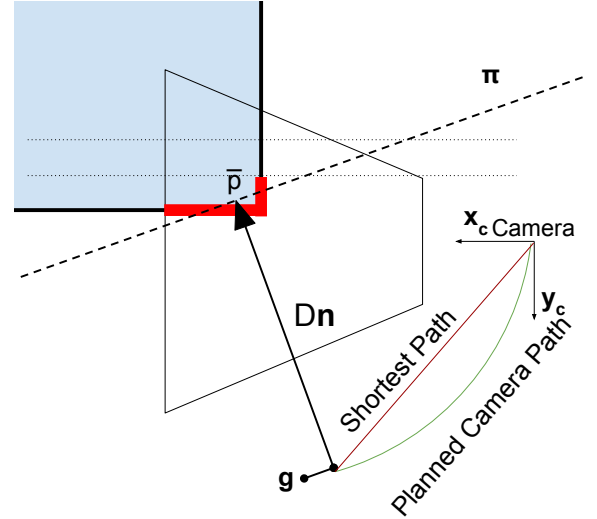


Fig. 4. Top-down view illustrating the computation of the next goal. We show here the situation for a corner section of the structure. The forward slice \mathcal{S} is highlighted in red and the corresponding best fit plane Π is shown as well.

pose is transformed into the global FoR to obtain the goal point g^g for the camera center C . We simplify the notation g^g to g in the following, where we work in the global reference frame.

B. Local path planning to the next goal

In order to move the camera center C to g while keeping it approximately at the desired distance D from the structure along the way, we use a local path planner based on potential fields [29], [30]. A potential function encoding the structure as obstacles in the neighborhood of the camera, as well as the goal g , is sampled in the form of a cost map on local 2D grid of size $2\rho \times 2\rho$ centered on the camera's current position, see Fig. 5. Assumption 2 guarantees that all the occupied cells in this cost map denote the structure itself. For k occupied cells centered at $\{x_j\}_{j=1}^k$, the potential function $N(x)$ is defined as

$$N(x) = \alpha \|x - g\|^2 + \sum_{j=1}^k I_j(x) d_j(x), \quad (1)$$

$$\text{with } d_j(x) = \frac{1}{\beta \|x - x_j\|}; I_j(x) = \begin{cases} 1 & \text{if } \|x - x_j\| \leq D \\ 0 & \text{otherwise,} \end{cases}$$

for some scalar parameters α, β . Here d_j is the repulsion from the j^{th} occupied cell, and is limited by I_j to a neighborhood of radius D around the cell. A path for the camera is obtained by following the negative gradient of N , i.e., $\dot{x} = -\nabla N(x)$. Denoting $J_x = \{j : I_j(x) = 1\}$ the occupied cells in the D -neighborhood of x , we have

$$-\nabla N(x) = 2\alpha(g - x) + \sum_{i \in J_x} \frac{1}{\beta \|x - x_i\|^3} (x - x_i). \quad (2)$$

Let $Q = \{x : J_x \neq \emptyset\}$ denote the region that is at distance at most D from the structure. Assuming a small value of β , the summation term in (2) is dominant whenever $x \in Q$ and pushes the path away from the structure. However, this term

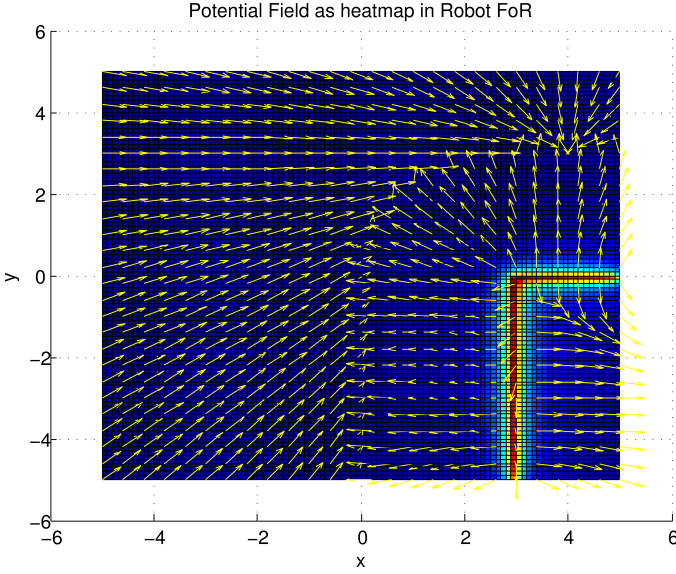


Fig. 5. The potential field for a goal at $(4, 3)$ with $D = 3$ is shown as a heat map and the corresponding gradient vectors are shown as a vector field.

vanishes as soon as $x \notin Q$. Then, assuming that the camera starts at x_0 on the boundary ∂Q of Q , it remains approximately on ∂Q if $g - x$ points toward the interior of Q . It is possible that this condition is not satisfied by the point g computed in the previous subsection, in which case we replace g by g_1 , which is obtained by selecting a new $goal = \bar{p}^c - D' \mathbf{n} + step \mathbf{r}$ for $D' < D$ such that this condition is satisfied. The path will then slide on ∂Q until it reaches its goal [31]. Finally, this path for the center C of the camera is used to compute a corresponding path for the center R of the robot that needs to be tracked using a platform specific controller.

C. Replanning due to the structure interfering

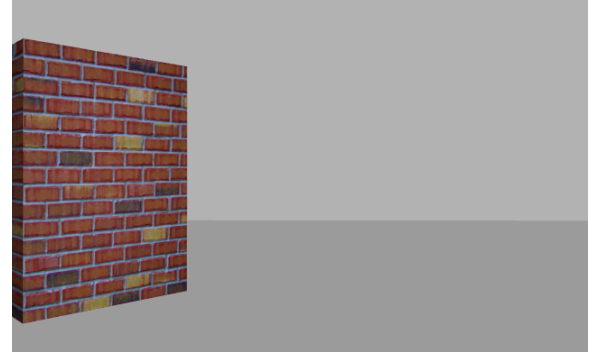
Assumption 2 guarantees that the robot can move sufficiently freely around the structure, but this does not prevent the structure itself from interfering with the path planned above. Consider the situation shown in Fig. 6a. The wall ahead of the robot does not fall into the FOV of the camera due to the limited horizontal angle of view, yet the robot should not approach this wall closer than a distance D . Hence, if the robot detects obstacles in its D -neighborhood, it is stopped at its current position and the yaw motion of the camera is used to scan ahead and face the new section of the structure. More precisely, as illustrated in Fig. 7, we use the costmap from the previous subsection to turn the camera to face along the direction from the robot center R to the first occupied cell in the D -neighborhood of the robot. The next goal is then recomputed using the newly captured point cloud.

D. End of the PE phase

The end of the PE phase is determined by monitoring the angle θ subtended by the robot's trajectory with respect to a reference point P inside the structure, see Fig. 8. The point P is chosen from the first depth image at the start of the PE phase. As shown in Fig. 8, the ray PR completes one full rotation in



(a)



(b)

Fig. 6. a) While the robot is following the structure, its forward facing sensors detect an obstacle ahead (robot configuration shown in faded colors). This obstacle is outside the field of view of the camera, shown in b). The position of the robot at the next waypoint along the new direction to explore, determined by using the arm to scan ahead, is shown in bright colors.

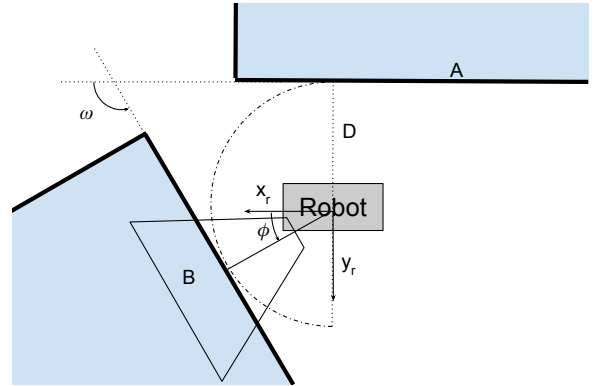


Fig. 7. When the robot is currently following section A of the structure, a later section B of the structure could interfere with the planned path. In general, the angle ω made by section B with respect to section A satisfies $\omega \in [0, \pi)$. Also, the two sections could be connected to form a non-convex corner.

the clockwise manner with respect to ray PO only at the end of the first pass. Note that θ need not monotonically increase and it depends on the geometry of the structure. But since by Initial Condition 1 there exists a plane NN' that initially separates the robot and the structure, θ can complete a full rotation only at the end of the first pass. Additionally, after θ completes a full rotation, we continue to follow the structure until the vSLAM module achieves a global loop closure.

Overall, our strategy attempts to maintain as much as

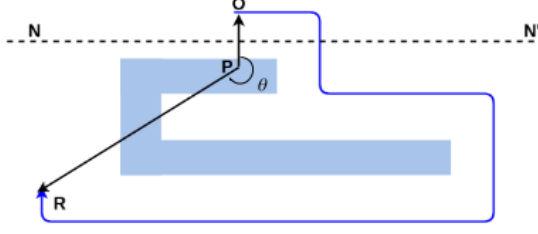


Fig. 8. The angle θ subtended by the path \widehat{OR} at a reference point P inside the structure can be used to detect the end of the PE phase.

possible a viewpoint orthogonal to the structure, even though it replans for a new goal according to Algorithm 1 only at discrete times. Note that only the computation of the next goal happens at discrete instants but the vSLAM module updates the model at a higher rate as per the capabilities of the hardware.

IV. COMPLETING THE MODEL

A. Determining flaws

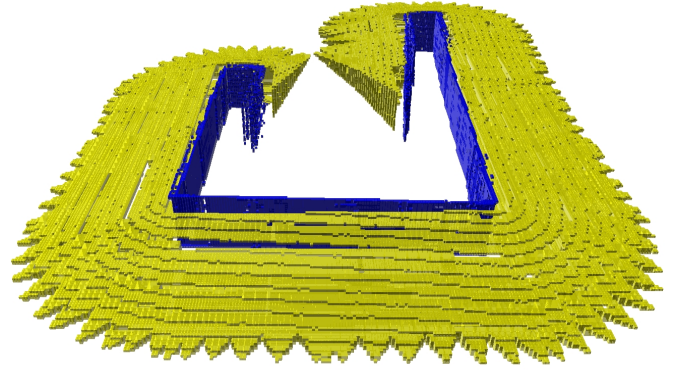
There are two possible types of flaws in the model obtained at the end of the PE phase. Type I flaws correspond to holes that are present in the already explored regions. As noted in Section II, these holes could be due to limitations of the sensor or local occlusions caused by small irregularities in the structure itself, and should be filled using a platform with a more appropriate reachable space, hence we do not consider them further. Type II flaws, called cavities in the following, correspond to regions that were missed during the PE phase, e.g., due to the situation depicted on Fig. 6, and will be filled during the CE phase. Our exploration policies only need the location of the entrances of the cavities and in this subsection we describe an algorithm to determine these locations in the model.

Our algorithm to determine the entrance to the cavities uses a voxel based 3D occupancy grid constructed from the global point cloud. We use the OctoMap [24] library to maintain this occupancy grid in a hierarchical tree data structure. Internally, the OctoMap library performs ray casting operations, labelling the occupancy measurement of each voxel along the line segment from the camera position to each point in the point cloud as *free* and the point itself as *occupied*. For this, we require the vSLAM module to provide the sequence of point clouds and associated camera positions used in assembling the current model. All voxels in the occupancy grid that are not labeled free or occupied are called *unknown*.

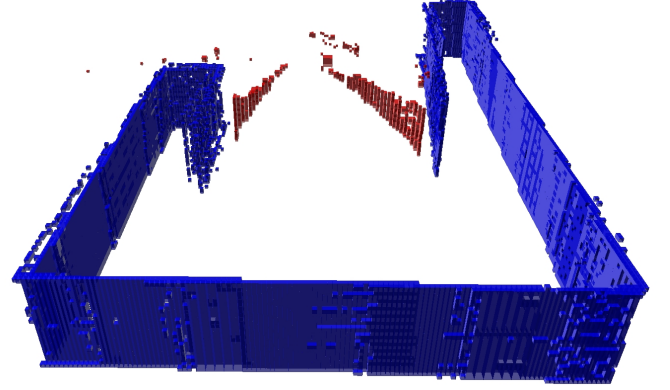
Using the constructed OctoMap, we compute a set of frontier voxels, whose definition is adapted from [6].

Definition 1: A *frontier voxel* is a free voxel with at least one neighboring unknown voxel.

Recall that the camera is constrained to move in a horizontal plane and the system continuously explored the structure in the clockwise sense during the PE phase. Consequently, point clouds captured before and after a cavity result in vertical planes of frontier voxels that border it and that we want to detect as entrance of the cavity for further exploration, see



(a) with frontier voxels



(b) with cavity entrance voxels

Fig. 9. a) The constructed OctoMap with occupied voxels shown in blue and frontier voxels shown in yellow. These yellow voxels form the boundary of the explored region and most of them lie along the top and bottom faces of the view frustums. b) The cavity entrance voxels are shown in red.

Fig. 9b. Therefore, we define *cavity entrance voxels* as frontier voxels satisfying two additional conditions. First, the normal to the frontier voxel, computed using the nearby frontier voxels [28], must approximately lie on the $x_g y_g$ plane. This serves to remove frontier voxels that lie along the top and bottom faces of the view frustums, See Fig. 9a. Second, Type I flaws can result in frontier voxels, which we want to exclude from cavity entrance voxels. Therefore, we require that the distance to the closest occupied voxel should be greater than some threshold d_0 , which can be chosen as a small fraction of the distance maintained from the structure, say $0.1 D$. As the number of voxels in a typical structure is very large, we do not perform this thresholding exactly but instead we use an estimate for the distance to the structure obtained from OctoMap. The hierarchical structure of OctoMap allows efficient multi-resolution queries, see Fig. 10, and thus we keep as cavity entrance voxels only those that are marked free at a resolution of approximately d_0 .

Finally, the cavity entrance voxels are clustered using an Euclidean clustering algorithm from PCL [27] and each cluster is referred to as a *cavity entrance*. Moreover, there could be some sparsely located cavity entrance voxels, which are removed by setting a minimum size for the cavity entrances.

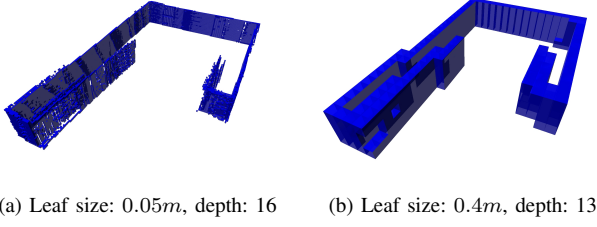


Fig. 10. OctoMap queried at depth level 16 and 13 respectively. In this paper, the value of D is 3m and the threshold d_0 is chosen as 0.4m.

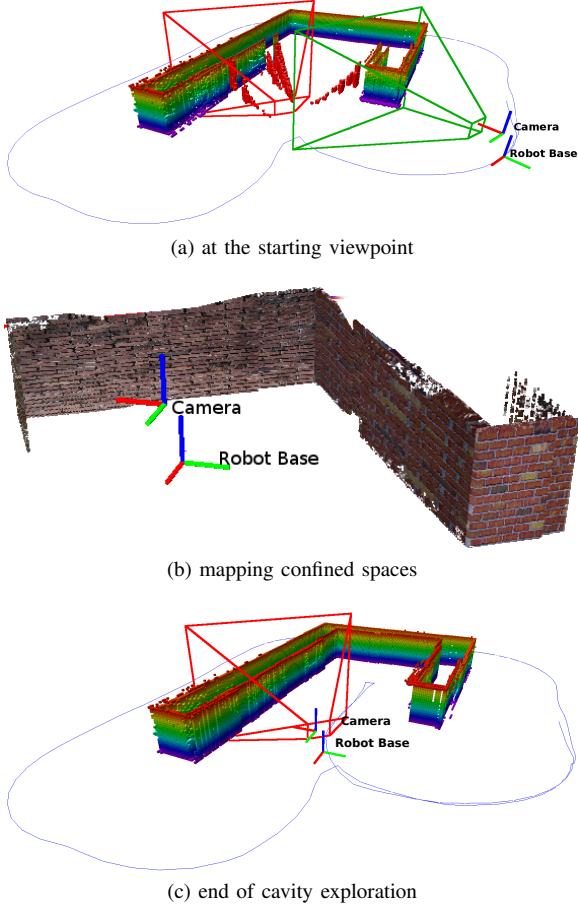


Fig. 11. Cavity exploration. a) The robot is at the starting viewpoint for exploring the cavity. The starting and ending viewpoints for the first cavity entrance are shown in green and red respectively. All the cavity entrance voxels are also shown. b) The extracted model showing CE in progress. c) The system detects the end of the cavity after coming close to the ending viewpoint. The blue line shows the trajectory followed by the robot.

B. Cavity Exploration

Once the cavity entrances have been determined, we can start the CE phase. We explore each detected cavity using an exploration strategy analogous to the PE phase. For this, we require a starting and ending viewpoint for each cavity entrance. We assume that there are no tunnels that go through the structure and that a robot entering a cavity can exit via the same location only. For a given cavity entrance, the starting viewpoint is chosen from the set of camera poses returned by the vSLAM module during the PE phase and such that the centroid of the cavity entrance lies within the view frustum.

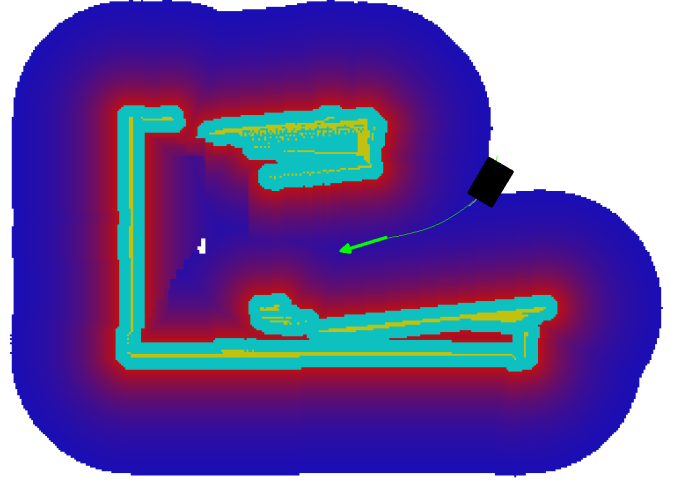


Fig. 12. Costmap at the beginning of the exploration of a cavity. The robot is shown as a black rectangle and the green arrow oriented along \mathbf{r} has its base at the computed next goal g such that $N(g)$ is a local minimum. The value of δ determined online is 2.0m while the value of D used in PE phase is 3.0m.

Additionally, the centroid should not be occluded by the structure from the camera position. From these camera poses, the one with the earliest timestamp τ_0 is chosen as starting viewpoint and we let τ_1 denote the largest timestamp. The ending viewpoint is chosen from the set of viewpoints from the PE phase as the one with the smallest timestamp τ_2 such that $\tau_2 > \tau_1$ and such that no cavity entrance voxel lies within the view frustum, see Fig. 11a.

The timestamps of the starting viewpoints of the cavity entrances are used to sort them in increasing order and each of the cavities is explored in sequence. A typical cavity has at least two cavity entrances bordering it and it is possible to have more cavity entrances if there is a row of pillars, for example. During the CE phase, if the centroid of a cavity entrance falls within the view frustum of the current camera position and is not occluded by the structure, we remove that cavity entrance from our list.

Exploring confined regions during the CE phase requires certain modifications to the PE policy. Recall that the system skipped the cavities during the PE phase as the robot came closer than a distance D from the structure. Therefore during the CE phase, only the region directly ahead of the robot and within a distance $\delta < D$ is checked for interference of the computed path with the structure. Moreover, our local path planner returns paths that maintain a distance δ from the structure, see Fig. 12. For this, using the notation of Sections III-A and III-B, we modify *goal* as $goal \leftarrow \bar{\mathbf{p}}^c - \delta \mathbf{n} + step \mathbf{r}$ and transform to the global FoR to obtain the new point g . The distance δ is chosen by starting from a small value and increasing it until we reach a local minimum of $N(g)$, where N is defined in (1).

The system detects that it has finished exploring the current cavity by monitoring the loop closures obtained by the vSLAM module. When the robot exits the current cavity, the point clouds captured by the camera correspond to parts of the structure that are already present in the model, see Fig. 11c.

TABLE I
SIMULATION RESULTS FOR DIFFERENT SIZES OF THE STRUCTURE AND
RANGE OF THE CAMERA

Model	Perimeter	Camera Range	Path Length
Small Γ	42m	4.5m	72.08m
Small Γ	42m	12.0m	53.79m
Large Γ	84m	4.5m	106.23m
Large \mathfrak{A}	94m	4.5m	179.65m

Consequently, these result in loop closures in the vSLAM module. We declare the cavity explored once the system receives a loop closure with a viewpoint at timestamp τ such that $\tau_2 < \tau < \tau_F$ where τ_F corresponds to the last viewpoint of the PE phase. Alternatively, the number of changes in the occupancy measurements of the OctoMap could be used to detect the end of the cavity, since the point clouds captured after exiting the cavity do not add new information to the OctoMap. But this solution tends to be less robust because the localization errors and sensor noise can induce a large number of changes even when the camera is viewing a region that is already present in the model.

V. SIMULATIONS AND RESULTS

We evaluate the performance of our policies via 3D simulations for different sizes of the structure, camera range values and localization accuracy levels for the robot. The implementation of our motion planning policies is integrated with the Robot Operating System (ROS) Navigation Stack [32], which is supported by many mobile ground robots. All the simulations are performed using the Gazebo simulator [15]. The vSLAM algorithm used is RTAB-Map [3].

The simulations are carried out with publicly available models of a Clearpath Husky A200 robot and a Kinect depth sensor whose range can be varied [33], see Fig. 2. A UR5 robotic arm is used to carry the sensor, but only yaw motions of the arm are allowed, as described in Section II. For illustration purposes, we consider artificial structures made of short wall-like segments. We refer to the structure used in most of the previous illustrations as the Small Γ model. The Large Γ model has the same shape as Small Γ but is twice the size. We also illustrate the effectiveness of our policy for a realistic model of a house, and compare its performance with that of the classic Frontier-Based Exploration (FBE) algorithm [6]. We have included a supplementary MP4 format video, which shows the simulation of a Husky robot following our policies for mapping the Small Γ model using a Kinect sensor with a range of 4.5m.

A. Structure Size and Camera Range

The relative size of the structure with respect to the range of the camera affects the trajectory determined by our algorithms. Fig. 13 shows simulation results for 4 scenarios. With a camera range of 4.5m, the Large Γ model is completely mapped at the end of the PE phase. For the Small Γ model a cavity remains, which is subsequently explored during the CE phase. Increasing the camera range to say 12m allows the Small Γ

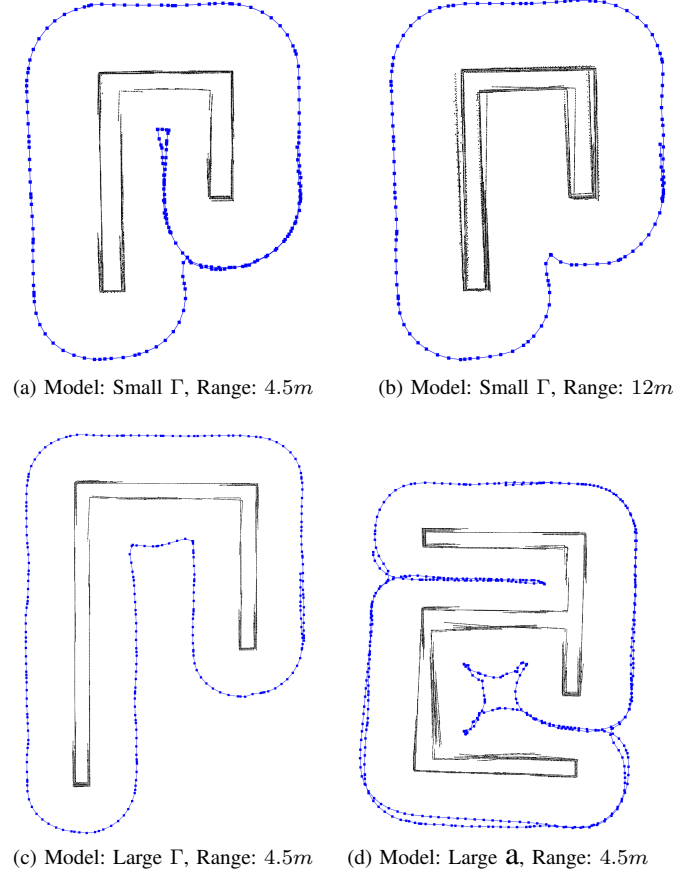


Fig. 13. The projection of the reconstructed model on the $\mathbf{x}_g \mathbf{y}_g$ plane is shown in black and the trajectory followed by the robot based on our policies is shown in blue.

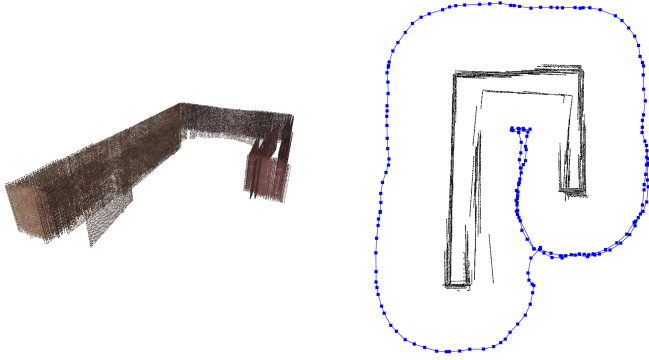
TABLE II
SIMULATION RESULTS FOR DIFFERENT LEVELS OF LOCALIZATION
ACCURACY

k	n_k	μ	σ	max
0.00	65520	0.05m	0.04m	0.20m
0.25	72636	0.08m	0.06m	0.27m
0.50	82728	0.11m	0.09m	0.40m
0.75	111321	0.16m	0.13m	0.94m

structure to be mapped at the end of the PE phase as well, see Fig. 13b. Fig. 13d shows that the robot following our policies is able to map large structures with multiple cavities of different sizes. Table I lists the path lengths obtained for the different test cases.

B. Localization accuracy

The Husky robot combines data from an Inertial Measurement Unit (IMU), a GPS module and wheel odometry to achieve a relatively small localization error overall. In order to evaluate the impact of localization accuracy on our algorithms, we simulate the effect of large wheel slippage by introducing a zero mean additive Gaussian white noise to each of the wheel encoder measurements, with a variance equal to $k(v_x + \omega_z)/2$, where v_x is the linear velocity of the robot, ω_z is its yaw rate and k is a proportionality constant, also called noise level



(a) Reconstructed model with $k = 0.75$

(b) Model: Small Γ , Range: 4.5m

Fig. 14. a) Large errors in localization results in poor alignment, although all portions of the structure have been captured in the model, see Fig. 15a for comparison. b) The projection of the reconstructed model on the $x_g y_g$ plane, when compared to Fig. 13a, shows the distortion introduced due to the noisy wheel odometry.

in the following. Increasing k results in a poorer alignment of the point clouds, but all portions of the structure, except the horizontal faces, are still captured in the reconstructed model, see Fig. 14. Note that our policies compute the next waypoint at discrete times and therefore assume that the drift in localization between waypoints is sufficiently small so that the robot reaches the next waypoint with the camera facing the structure.

We use the CloudCompare [34] software to compute the distortion in the reconstructed model C_k , for a noise level k , with respect to a reference point cloud C_R which is generated using a different mobile platform with almost perfect localization. First, we register C_k to C_R using an Iterative Closest Point (ICP) algorithm [35]. We then define for every point in C_k , its error to be the distance to the nearest neighbor in C_R . Table II lists the simulation results for mapping the Small Γ model with different noise levels k , where n_k is the number of points in C_k and μ, σ, \max are respectively the mean, standard deviation and maximum value of the errors of all points in C_k . The table indicates that both the mean and standard deviation of the errors increase with the noise level.

C. Comparing with Frontier-based Exploration

The Frontier Exploration [36] package available in ROS relies on a 2D LIDAR to build an occupancy grid that is used to compute the frontiers. The package requires the user to define a 2D polygon that encloses the structure. The algorithm then explores until there are no more frontiers inside the user-defined polygon. In comparison to the FBE algorithm, our algorithms

- do not require a user defined bounding polygon;
- maintain as much as possible a fixed distance from the structure (during the PE phase), thereby ensuring that all portions up to a height of H_{\max} are mapped;
- consistently explore the structure in the clockwise direction, which can be important from a user perspective to understand the behavior of the robot. On the other hand,

TABLE III
COMPARISON BETWEEN OUR POLICY AND FRONTIER BASED EXPLORATION

		Proposed Policy	FBE
Small Γ	Path Length	72.08m	49.78m
	Unique closest point set size	6,063	5,398
	Mean Error	0.05m	0.12m
House	Path Length	59.89m	47.55m
	Unique closest point set size	9,182	7,402
	Mean Error	0.05m	0.12m

the trajectory prescribed by the FBE algorithm depends on the size of the user-defined bounding polygon. A large bounding polygon will cause the robot to explore areas far away from the structure and will possibly not maintain a fixed direction of exploration.

- produces the same path every time for a given structure whereas the path computed by the FBE algorithm could differ greatly between two trials. Also, the robot often gets stuck while using the FBE algorithm as the computed waypoints are often too close to the structure.

In order to get a quantitative measure of the structure coverage, we use again the CloudCompare software to compute essentially the projection of the reconstructed model C on the reference point cloud C_R . Namely, for each point in C we compute the closest point in C_R . Note that multiple points in C can have the same closest point in C_R . In this case, we remove these duplicate points to obtain the *unique closest point set*. Then, as long as the reconstructed model aligns relatively well with the reference model, the cardinality of the unique closest point set is taken as our estimate of the structure coverage. Table III compares the level of structure coverage achieved by our policies and FBE with a camera range of 4.5m for two of the environments considered. For the Small Γ model, our reference point cloud has 6,116 points with a minimum distance of 0.1m between points. For the House model, our reference point cloud has 10,889 points with a minimum distance of 0.1m between points. Since the height of the House model is more than H_{\max} , we only take the portion of the reconstructed model up to the height H_{\max} for computing the structure coverage and mean error for the two algorithms.

Table III shows that our policies achieve a higher level of structure coverage than FBE for the environments considered and our proposed coverage metric. Note also that the smooth trajectory prescribed by our policies is beneficial to the vSLAM module to achieve a better alignment and a lower value for the mean error in the reconstructed model, especially if the robot localization accuracy is poor. A visual inspection of Fig. 15a and Fig. 15b shows the improvement in performance of the vSLAM module when using our algorithms compared to FBE.

VI. CONCLUSION AND FUTURE WORK

This paper presents novel motion planning policies that guide a mobile ground robot carrying a depth sensor to

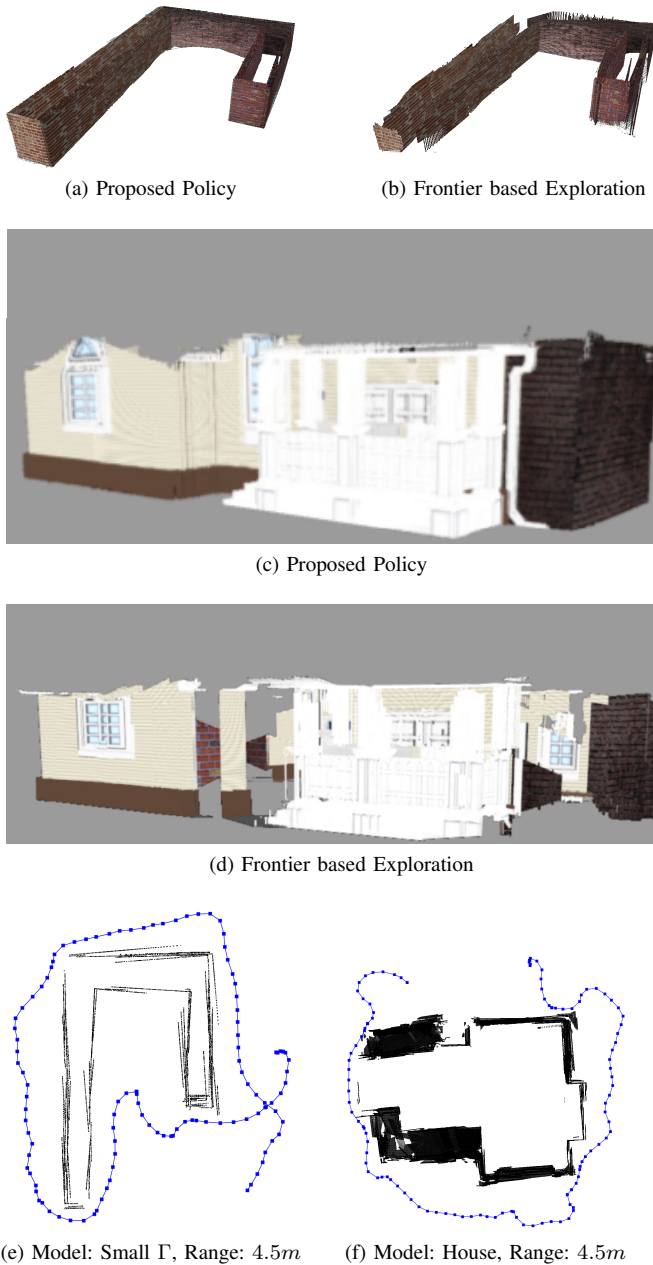


Fig. 15. (a-d) Comparison of the reconstructed model using our policies and FBE. (e,f) The trajectory prescribed by FBE for the Small Γ and House structure is shown in blue.

autonomously explore the visible portion of a bounded three-dimensional structure. The proposed policies do not assume any prior information about the size or geometry of the structure. Coupled with state-of-art vSLAM systems, our strategies are able to achieve high coverage in the reconstructed model, given the physical limitations of the platform. We have illustrated the efficacy of our approach via 3D simulations for different structure sizes, camera range and localization accuracy. In addition, a comparison of our policies with the classic frontier based exploration algorithms clearly shows the improvement in performance for a realistic structure such as a house.

This work opens interesting questions such as combined

vehicle and 6-DOF arm motion planning to map fine structural details. Extending our algorithms to aerial platforms and hybrid teams of robots, for example, would also allow for autonomous inspection of tall structures such as wind turbines or telecom towers.

REFERENCES

- [1] M. Jacobi, "Autonomous inspection of underwater structures," *Robotics and Autonomous Systems*, vol. 67, no. C, pp. 80–86, May 2015, special issue on Advances in Autonomous Underwater Robotics.
- [2] S. F. El-Hakim, J. A. Beraldin, M. Picard, and G. Godin, "Detailed 3D reconstruction of large-scale heritage sites with integrated techniques," *IEEE Computer Graphics and Applications*, vol. 24, no. 3, pp. 21–29, May-June 2004.
- [3] M. Labbe and F. Michaud, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, September 2014.
- [4] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, February 2014.
- [5] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, August 1988.
- [6] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, ser. CIRA '97, 1997, pp. 146–151.
- [7] S. Kriegl, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects," *Journal of Real-Time Image Processing*, pp. 611–631, December 2015.
- [8] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3D object models using next best view manipulation planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 5031–5037.
- [9] J. Lin, K. Han, and M. Golparvar-Fard, "A framework for model-driven acquisition and analytics of visual data using UAVs for automated construction progress monitoring," in *Computing in Civil Engineering*, Austin, Texas, June 2015, pp. 156–164.
- [10] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, May 2015, pp. 6423–6430.
- [11] E. Acar, H. Choset, and J. Lee, "Sensor-based coverage with extended range detectors," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 189–198, Feb 2006.
- [12] R. Lim, H. M. La, and W. Sheng, "A robotic crack inspection and mapping system for bridge deck maintenance," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 367–378, April 2014.
- [13] B. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures," in *International Conference on Automated Planning and Scheduling (ICAPS)*, Sao Paulo, Brazil, June 2012, pp. 29–37.
- [14] W. Sheng, H. Chen, and N. Xi, "Navigating a miniature crawler robot for engineered structure inspection," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 2, pp. 368–373, April 2008.
- [15] Gazebo robot simulator. <http://gazebo.org/>. Accessed: 2015-12-29.
- [16] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, "Building Rome on a cloudless day," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV'10, 2010, pp. 368–381.
- [17] C. Wu. (2011) VisualSFM: A visual structure from motion system. <http://ccwu.me/vsfm/>. Accessed: 2015-12-27.
- [18] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, Aug 2010.
- [19] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 3607–3613.

- [20] R. Shade and P. Newman, "Choosing where to go: Complete 3D exploration with stereo," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 2806–2811.
- [21] S. Shen, N. Michael, and V. Kumar, "Autonomous indoor 3D exploration with a micro-aerial vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, May 2012, pp. 9–15.
- [22] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, May 2015, pp. 1071–1078.
- [23] N. Atanasov, J. Le Ny, K. Daniilidis, and G. J. Pappas, "Decentralized active information acquisition: Theory and application to multi-robot SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, May 2015, pp. 4775–4782.
- [24] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, April 2013.
- [25] S. Zhou, J. Xi, M. W. McDaniel, T. Nishihata, P. Salesses, and K. Iagnemma, "Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain," *Journal of Field Robotics*, vol. 29, no. 2, pp. 277–297, 2012.
- [26] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," *Advanced Robotics*, vol. 27, pp. 459–468, 2013.
- [27] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 1–4.
- [28] N. J. Mitra, A. Nguyen, and L. Guibas, "Estimating surface normals in noisy point cloud data," *International Journal of Computational Geometry & Applications*, vol. 14, no. 04n05, pp. 261–276, 2004.
- [29] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [30] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.
- [31] J. Cortes, "Discontinuous dynamical systems," *IEEE Control Systems Magazine*, vol. 28, no. 3, pp. 36–73, June 2008.
- [32] ROS navigation stack. <http://wiki.ros.org/navigation>. Accessed: 2015-11-26.
- [33] Gazebo plugins. http://wiki.ros.org/gazebo_plugins. Accessed: 2015-12-22.
- [34] CloudCompare (version 2.6.0) [GPL software]. (2016). Retrieved from <http://www.cloudcompare.org/>.
- [35] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, 2001*, Quebec City, Canada, May 2001, pp. 145–152.
- [36] ROS frontier exploration. http://wiki.ros.org/frontier_exploration. Accessed: 2015-11-26.